SELÇUK ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

3302659-3312659
GÖRÜNTÜ İŞLEME(YM)

PROJE 3

DERSİN HOCASI

Dr. Kürşad UÇAR

Enes ÇAKIR
203312011

Proje Tarihi	Rapor Teslim Tarihi	Proje Notu
08/05/2025	15/05/2025	

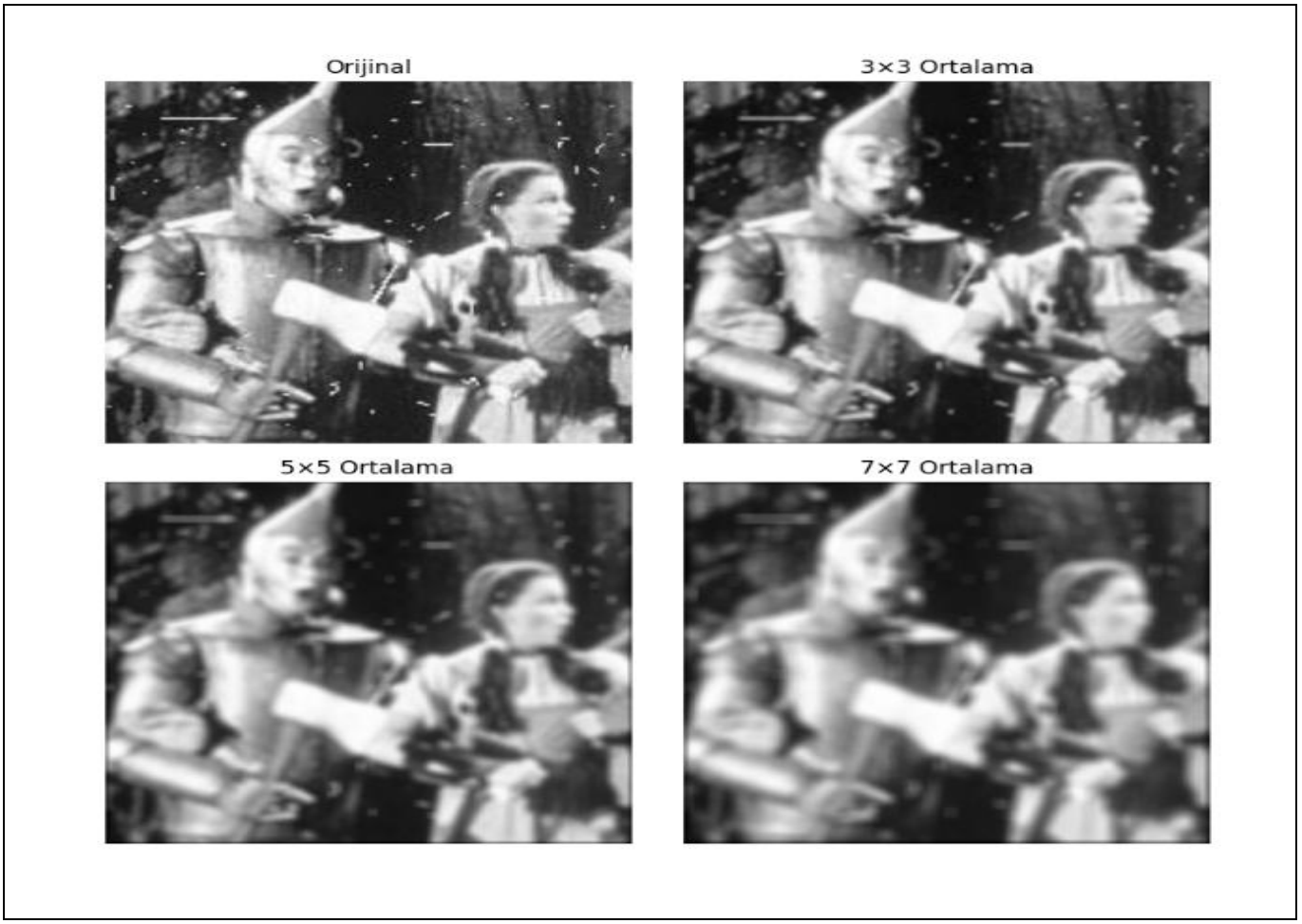
1. PROJENİN AMACI

Bu projenin amacı, dijital görüntü işleme tekniklerini kullanarak çeşitli görüntü iyileştirme ve analiz yöntemlerini uygulamaktır. Projede üç temel konu başlığı ele alınmaktadır: alçak geçiren filtreleme, yüksek geçiren filtreleme ve histogram eşitleme. İlk olarak, *gurultulugoruntu.pgm* dosyası üzerinde ortalama ve median filtreleme yöntemleri uygulanarak gürültü azaltma işlemi gerçekleştirilecektir. Bu filtreleme işlemleri farklı boyutlardaki çekirdekler (3x3, 5x5, 7x7) ile test edilerek yöntemlerin etkililiği karşılaştırılacak ve sonuçları yorumlanacaktır. Ardından, *santractahtasi.pgm* görüntüsü üzerinde Sobel ve Laplace gibi yüksek geçiren filtreler kullanılarak kenar belirginleştirme yapılacak ve görüntüdeki detayların keskinleştirilmesi sağlanacaktır. Son olarak, renkli bir görüntü olan *cicek.ppm* dosyasında histogram eşitleme işlemi gerçekleştirilerek her bir renk kanalında (kırmızı, yeşil, mavi) kontrast artırımı sağlanacaktır.

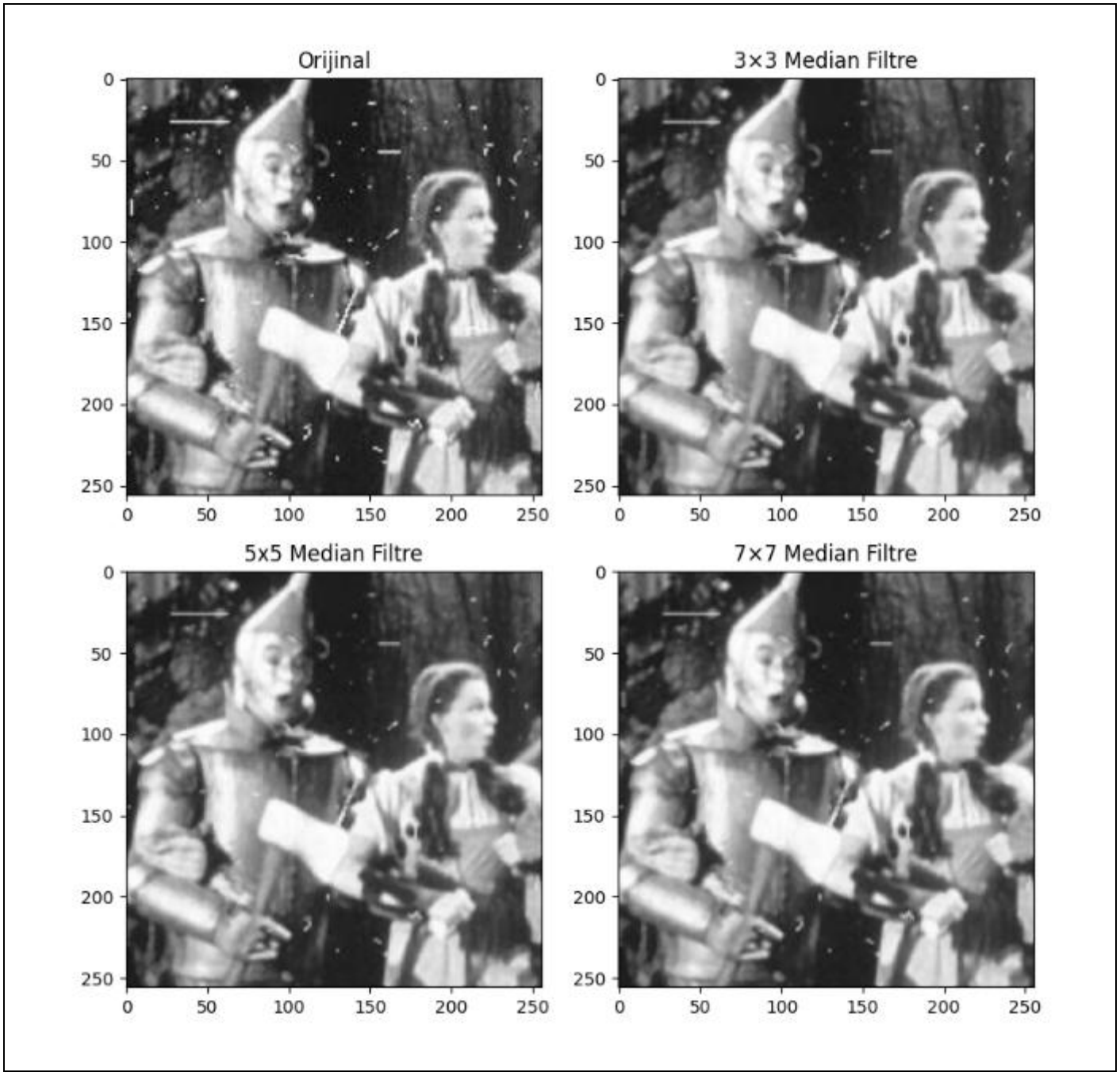
2. TEST SONUÇLARI

2.1 Görüntünde Alçak Geçiren Filtreleme Sonuçları

Ortalama Filtreleme: Ortalama filtreleme, görüntüdeki her pikselin değerini, etrafındaki komşu piksellerin ortalamasını alarak yeniden hesaplayan bir alçak geçiren filtredir. Bu yöntemde, seçilen pencere boyutuna (örneğin 3x3, 5x5, 7x7) göre merkez pikselin çevresindeki tüm piksellerin yoğunluk değerleri toplanır ve bu değerlerin aritmetik ortalaması merkez pikselin yeni değeri olarak atanır. Gürültülü bir görüntüye uygulandığında, bu filtre yüksek frekanslı bileşenleri (ani değişimleri) bastırarak görüntüyü yumuşatır ve rastgele oluşmuş küçük parazitleri azaltır. Ancak, ortalama filtre kenar detaylarını da bulanıklaştırabileceğinden, keskinlik kaybı yaşanabilir.

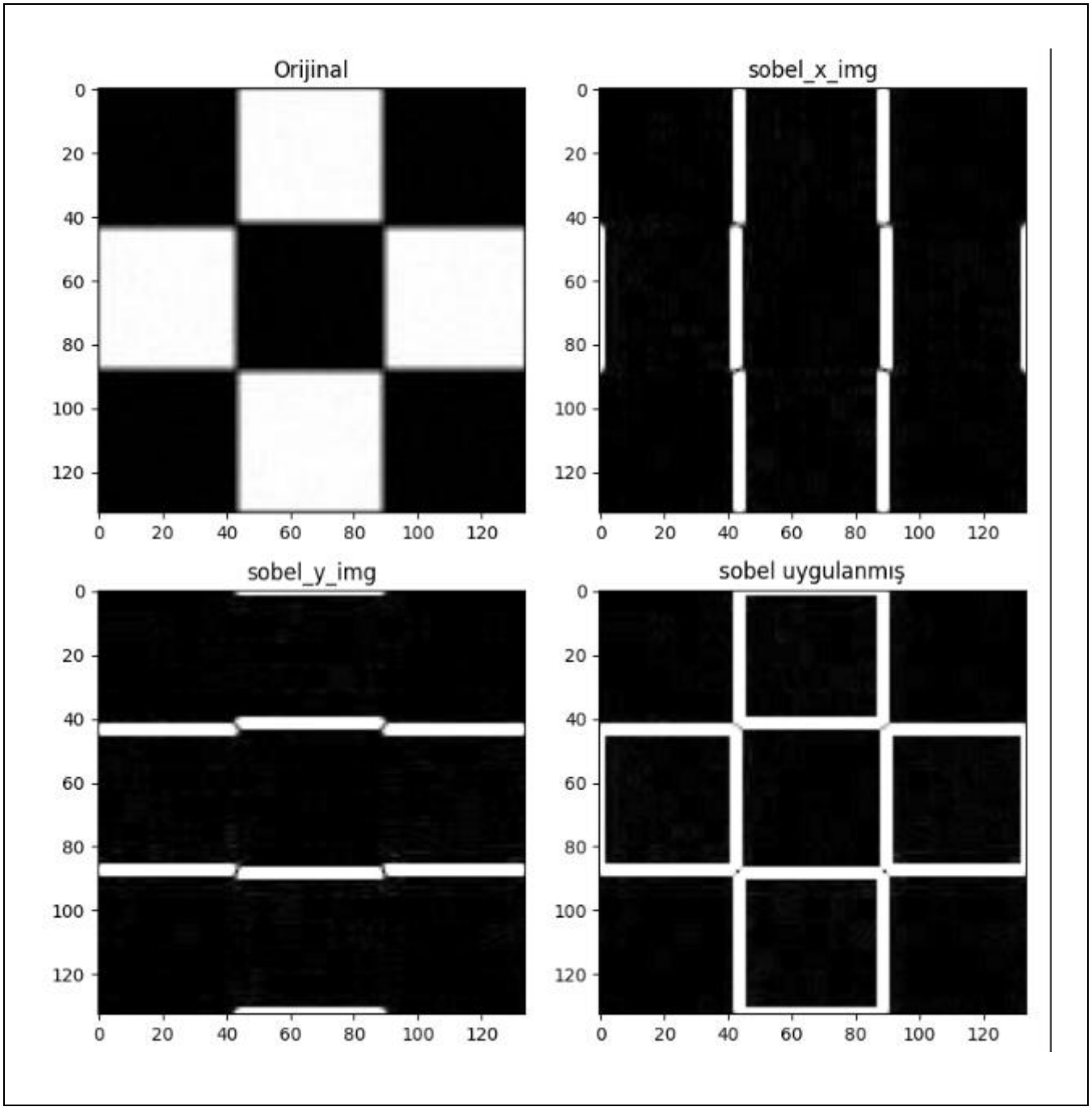


Median(Ortanca)Filtreleme: Median filtreleme ise her pikselin değerini, komşuluk penceresindeki piksellerin sıralanarak ortanca (median) değeri ile değiştirir. Median filtre, ortalama filtreye göre kenarları daha iyi korur ve görüntüdeki detayların kaybolmasını daha az etkiler. Filtre boyutu arttıkça yumuşatma etkisi artar ancak çok büyük pencereler görüntüyü aşırı düzleştirebilir. Sonuç olarak median filtre, keskinlikten fazla ödün vermeden gürültü azaltımı sağlar.

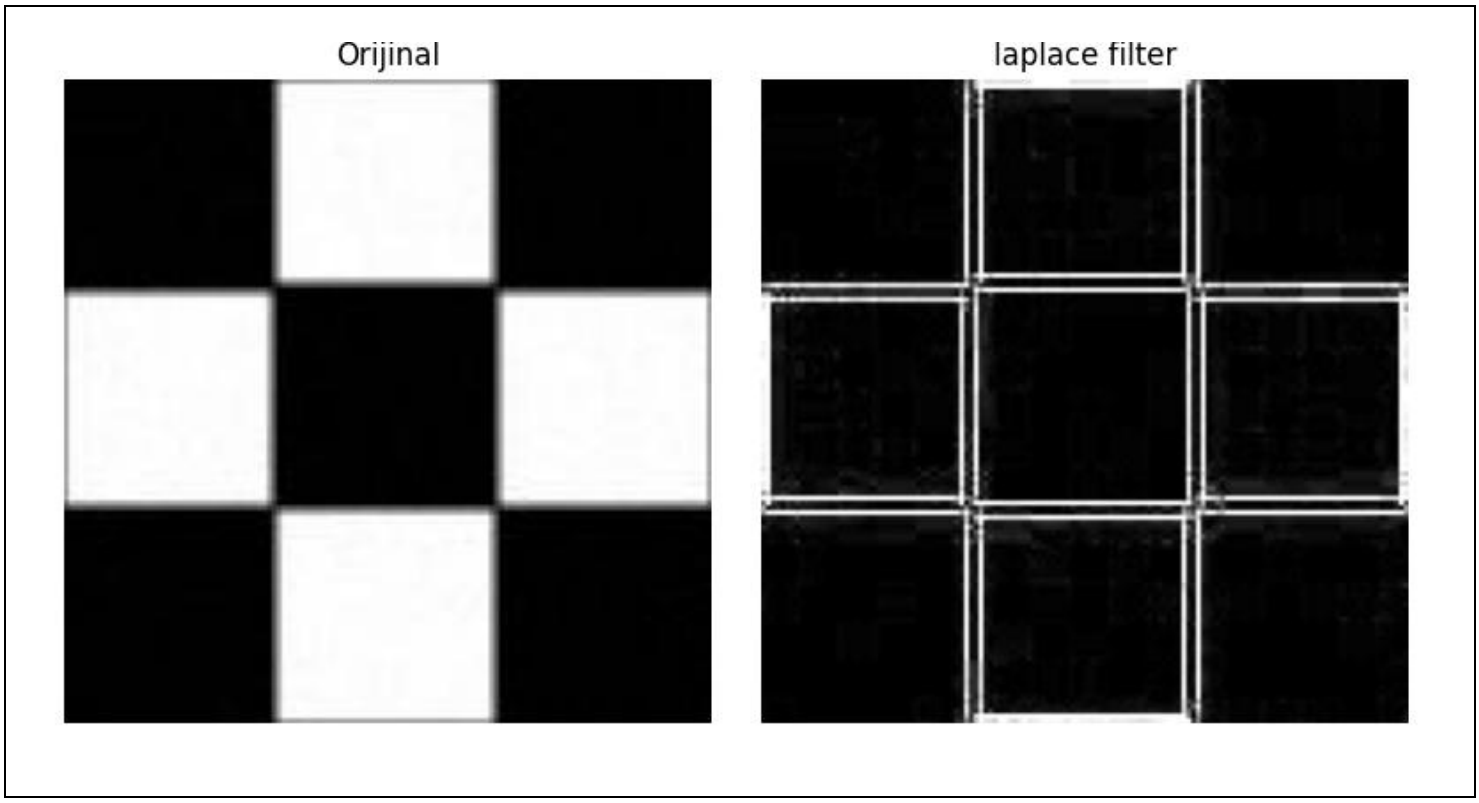


2.2 Görüntüde Yüksek Geçiren Filtreleme Sonuçları

Sobel Filtresi: Sobel filtresi, bir görüntüdeki kenarları belirlemek amacıyla kullanılan bir yüksek geçiren filtredir. Bu yöntem hem yatay hem de düşey yönde gradyan (parlaklık değişimi) hesaplayarak kenarları tespit eder. Görüntü üzerinde, belirli bir çekirdek (kernel) matrisi kullanılarak piksellerin x ve y yönlerindeki türevleri alınır. Bu türevler, kenarın şiddetini ve yönünü belirlemeye yardımcı olur. Sobel filtresi uygulandığında, kenarların bulunduğu bölgelerde yüksek gradyan değerleri üretilir ve bu sayede görüntüdeki nesnelerin sınırları daha belirgin hale gelir. Gürültüye karşı Laplace filtresine göre daha dayanıklıdır çünkü gradyan hesaplamalarında bir miktar yumuşatma içerir.

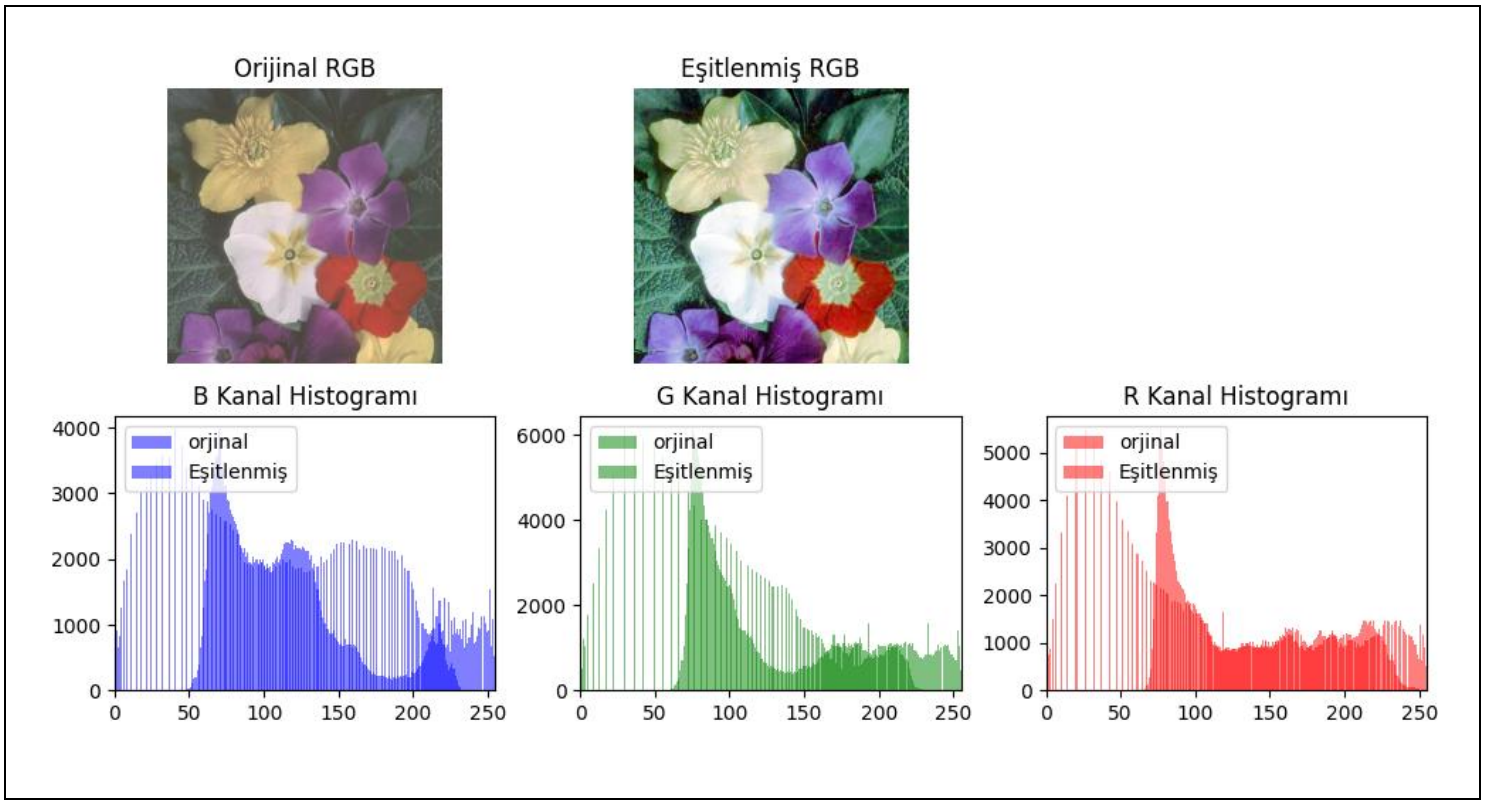


Laplace Filtresi: Laplace filtresi, bir görüntüdeki ikinci türev bilgilerini kullanarak kenar belirlemesi yapan bir başka yüksek geçiren filtredir. Bu filtre hem yatay hem de düşey doğrultuda aynı anda değişimi analiz eder ve keskin parlaklık geçişlerinin olduğu bölgelerde yüksek değerler üretir. Laplace filtresi uygulandığında, görüntüdeki kenarlar daha ince çizgiler halinde ortaya çıkar ve kenar keskinliği artırılır. Ancak, bu yöntem gürültüye karşı hassastır ve küçük gürültüler bile kenar gibi algılanabilir. Bu nedenle uygulamadan önce alçak geçiren bir filtreyle gürültü azaltılması önerilir. Sonuç olarak Laplace filtresi, detayları ön plana çıkarmak için güçlü bir araçtır ancak dikkatli kullanılmalıdır.



2.1 Görüntünde Histogram Eşitleme (Histogram Equalization) Sonuçları

Histogram Eşitleme (Histogram Equalization): Histogram eşitleme, bir görüntünün kontrastını artırmak amacıyla kullanılan yaygın bir iyileştirme tekniğidir. Bu yöntem, görüntüdeki parlaklık değerlerinin dağılımını daha dengeli hâle getirerek koyu ve açık alanlar arasındaki farkları belirginleştirir. Renkli görüntülerde bu işlem her bir renk kanalı (kırmızı, yeşil, mavi) için ayrı ayrı uygulanır. Her kanalın parlaklık histogramı analiz edilir, kümülatif dağılım fonksiyonu (CDF) hesaplanarak mevcut piksel değerleri yeniden ölçeklendirilir. Bu işlem sonucunda her bir kanal daha geniş bir ton aralığına yayılmış olur ve genel görüntü daha canlı, detayları daha belirgin hale gelir. Histogram eşitleme, özellikle düşük kontrastlı veya dengesiz aydınlatılmış görüntülerde önemli iyileştirmeler sağlar. Ancak renkli görüntülerde her kanalı ayrı eşitlemek, bazen renk dengesinde hafif bozulmalara yol açabilir. Buna rağmen, görüntü kalitesinin genel olarak artırılmasında etkili bir yöntemdir.



3.1 Görüntüyü Alçak Geçiren Filtreleme (ortalama değer ve Median) Kodları

```
#ortalamadeğer
import cv2
import numpy as np
import matplotlib.pyplot as plt

in_image = cv2.imread('gurultulugoruntu.jpg', cv2.IMREAD_GRAYSCALE)

def fonksiyon_mean(image, kernel_size):
    pad = kernel_size // 2
    rows, cols = image.shape
    padded = np.zeros((rows + 2 * pad, cols + 2 * pad), dtype=image.dtype)
    padded[pad:pad + rows, pad:pad + cols] = image
    output = np.zeros_like(image)
    for i in range(rows):
        for j in range(cols):
            region = padded[i: i + kernel_size, j: j + kernel_size]
            total = 0
            for m in range(kernel_size):
                for n in range(kernel_size):
                    total += int(region[m, n])
            count = kernel_size * kernel_size #eleman sayısı
            mean_val = total // count #ortalama_deger
            output[i, j] = mean_val
    return output

out3 = fonksiyon_mean(in_image, 3)
out5 = fonksiyon_mean(in_image, 5)
out7 = fonksiyon_mean(in_image, 7)

plt.figure(figsize=(8, 8))

plt.subplot(2, 2, 1)
plt.imshow(in_image, cmap='gray')
plt.title("Orijinal")
plt.axis('off')

plt.subplot(2, 2, 2)
plt.imshow(out3, cmap='gray')
plt.title("3×3 Ortalama")
plt.axis('off')

plt.subplot(2, 2, 3)
plt.imshow(out5, cmap='gray')
plt.title("5×5 Ortalama")
plt.axis('off')

plt.subplot(2, 2, 4)
plt.imshow(out7, cmap='gray')
plt.title("7×7 Ortalama")
plt.axis('off')

plt.tight_layout()
plt.show()
```

```
#mediandeğer
import cv2
import numpy as np
import matplotlib.pyplot as plt

in_image = cv2.imread('gurultulugoruntu.jpg', cv2.IMREAD_GRAYSCALE)

def fonksiyon_median(image, kernel_size):
    pad = kernel_size // 2
    rows, cols = image.shape
    padded = np.zeros((rows + 2 * pad, cols + 2 * pad), dtype=image.dtype)
    padded[pad:pad + rows, pad:pad + cols] = image
    output = np.zeros_like(image)
    for i in range(rows):
        for j in range(cols):
            region = padded[i: i + kernel_size, j: j + kernel_size]
            med_val = np.median(region)
            output[i, j] = int(med_val)
    return output

out_med3 = fonksiyon_median(in_image, 3)
out_med5 = fonksiyon_median(in_image, 5)
out_med7 = fonksiyon_median(in_image, 7)

plt.figure(figsize=(8, 8))
plt.subplot(2, 2, 1)
plt.imshow(in_image, cmap='gray')
plt.title("Orijinal")

plt.subplot(2, 2, 2)
plt.imshow(out_med3, cmap='gray')
plt.title("3×3 Median Filtre")

plt.subplot(2, 2, 3)
plt.imshow(out_med3, cmap='gray')
plt.title("5x5 Median Filtre")

plt.subplot(2, 2, 4)
plt.imshow(out_med3, cmap='gray')
plt.title("7×7 Median Filtre")

plt.tight_layout()
plt.show()
```

3.2 Görüntüde Yüksek Geçiren Filtreleme (Sobel ve Laplace) kodları

```
#sobel
import cv2
import numpy as np
import matplotlib.pyplot as plt

def sobel_filter(img):
    #sobel_operators
    sobel_x = np.array([[-1, 0, 1],
                        [-2, 0, 2],
                        [-1, 0, 1]], dtype=np.float32)
    sobel_y = np.array([[-1, -2, -1],
                        [ 0,  0,  0],
                        [ 1,  2,  1]], dtype=np.float32)
```

```
    # Padding -- kenar taşmasını önlemek için padding işlemini yapıyoruz
    pad = 1
    padded = np.pad(img.astype(np.float32), pad, mode='constant', constant_values=0)


    h, w = img.shape
    gx = np.zeros((h, w), dtype=np.float32)
    gy = np.zeros((h, w), dtype=np.float32)


    for i in range(h):
        for j in range(w):
            window = padded[i:i+3, j:j+3]
            gx[i, j] = np.sum(window * sobel_x)
            gy[i, j] = np.sum(window * sobel_y)

    gx = np.clip(np.abs(gx), 0, 255).astype(np.uint8)
    gy = np.clip(np.abs(gy), 0, 255).astype(np.uint8)

    magnitude = np.sqrt(gx.astype(np.float32)**2 + gy.astype(np.float32)**2)
    magnitude = np.clip(magnitude, 0, 255).astype(np.uint8)

    return gx, gy, magnitude

def main():
    in_image = cv2.imread('santractahtasi.jpg', cv2.IMREAD_GRAYSCALE)

    sobel_x_img, sobel_y_img, sobel_combined = sobel_filter(in_image)

    plt.figure(figsize=(8, 8))
    plt.subplot(2, 2, 1)
    plt.imshow(in_image, cmap='gray')
    plt.title("Orijinal")

    plt.subplot(2, 2, 2)
    plt.imshow(sobel_x_img, cmap='gray')
    plt.title("sobel_x_img")

    plt.subplot(2, 2, 3)
    plt.imshow(sobel_y_img, cmap='gray')
    plt.title("sobel_y_img")

    plt.subplot(2, 2, 4)
    plt.imshow(sobel_combined, cmap='gray')
    plt.title("sobel uygulanmış")

    plt.tight_layout()
    plt.show()

if __name__ == '__main__':
    main()


#laplace
import cv2
import numpy as np
```

```
import matplotlib.pyplot as plt


def laplace_filter(img):

    # kernel 1
    kernel = np.array([[1, 1, 1],
                       [1, -8, 1],
                       [1, 1, 1]], dtype=np.float32)
    pad = 1
    padded = np.pad(img.astype(np.float32), pad, mode='constant',
constant_values=0)

    h, w = img.shape
    img_zeros = np.zeros((h, w), dtype=np.float32)

    for i in range(h):
        for j in range(w):
            window = padded[i:i+3, j:j+3]
            img_zeros[i, j] = np.sum(window * kernel)

    img_zeros = np.clip(np.abs(img_zeros), 0, 255).astype(np.uint8)

    return img_zeros

def main():

    in_image = cv2.imread('santractahtasi.jpg', cv2.IMREAD_GRAYSCALE)

    out_image = laplace_filter(in_image)

    plt.figure(figsize=(8, 4))
    plt.subplot(1, 2, 1)
    plt.imshow(in_image, cmap='gray')
    plt.title("Orijinal")
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(out_image, cmap='gray')
    plt.title("laplace filter")
    plt.axis('off')

    plt.tight_layout()
    plt.show()

if __name__ == '__main__':
    main()
```

3.3 Görüntünde Histogram Eşitleme (Histogram Equalization) Kodları

```
#hazır fonksiyon kullanımı ile ilgili bilgi verilmediği için bu kodda kullandım
import cv2
import numpy as np
import matplotlib.pyplot as plt


# 1. Görüntüyü yükle (BGR formatında)
img_bgr = cv2.imread('cicek.jpeg', cv2.IMREAD_COLOR_BGR)

# 2. Kanallarına ayır
b, g, r = cv2.split(img_bgr)

# 3. Her kanal için histogram eşitleme
b_eq = cv2.equalizeHist(b) #Blue
g_eq = cv2.equalizeHist(g) #Green
r_eq = cv2.equalizeHist(r) #Red #EQ=EQUALIZATION KISALTMASI

# 4. Eşitlenmiş kanalları birleştir
img_eq_bgr = cv2.merge((b_eq, g_eq, r_eq))

img_rgb       = cv2.cvtColor(img_bgr,    cv2.COLOR_BGR2RGB)
img_eq_rgb    = cv2.cvtColor(img_eq_bgr, cv2.COLOR_BGR2RGB)

plt.figure(figsize=(10, 5))

plt.subplot(2, 3, 1)
plt.imshow(img_rgb)
plt.title('Orijinal RGB')
plt.axis('off')

plt.subplot(2, 3, 2)
plt.imshow(img_eq_rgb)
plt.title('Eşitlenmiş RGB')
plt.axis('off') #axis off görsellerin çözünürlüğü göstermek için kullanılıyor

# 7. Her kanalın histogramını da çiz
channels = ['B', 'G', 'R']
orig_channels = [b, g, r]
eq_channels   = [b_eq, g_eq, r_eq]
colors = ['blue', 'green', 'red']

for i, ch in enumerate(channels):
    plt.subplot(2, 3, 4 + i)
    plt.hist(orig_channels[i].ravel(), bins=256, range=(0,255), color=colors[i],
             alpha=0.5, label='orjinal')
    plt.hist(eq_channels[i].ravel(),   bins=256, range=(0,255), color=colors[i],
             alpha=0.5, label='Eşitlenmiş')
    plt.title(f'{ch} Kanal Histogramı')
    plt.xlim([0,255])
    plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```